

AFRL-IF-WP-TM-2003-1501

**AN OPEN LOGICAL PROGRAMMING
ENVIRONMENT**

**A Practical Framework for Sharing Formal
Models**

**Robert L. Constable
Christoph Kreitz**

**Cornell University
Department of Computer Science
4130 Upson Hall
Ithaca, NY 14853**



DECEMBER 2002

Final Report for 01 February 1998 – 31 December 2002

Approved for public release; distribution is unlimited.


**INFORMATION DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7334**

NOTICE


USING GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA INCLUDED IN THIS DOCUMENT FOR ANY PURPOSE OTHER THAN GOVERNMENT PROCUREMENT DOES NOT IN ANY WAY OBLIGATE THE U.S. GOVERNMENT. THE FACT THAT THE GOVERNMENT FORMULATED OR SUPPLIED THE DRAWINGS, SPECIFICATIONS, OR OTHER DATA DOES NOT LICENSE THE HOLDER OR ANY OTHER PERSON OR CORPORATION; OR CONVEY AND RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY RELATE TO THEM.

THIS REPORT HAS BEEN REVIEWED BY THE OFFICE OF PUBLIC AFFAIRS (ASC/PA) AND IS RELEASABLE TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). AT NTIS, IT WILL BE AVAILABLE TO THE GENERAL PUBLIC, INCLUDING FOREIGN NATIONS.

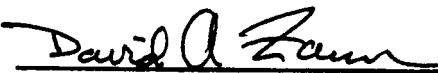
THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.



RONALD SZKODY
PCES Program Manager,
Advanced Architecture &
Integration Branch



STEPHEN L. BENNING
Team Leader
Advanced Architecture &
Integration Branch



DAVID A. ZANN, Chief
Advanced Architecture & Integration Branch
Information Systems Division

COPIES OF THIS REPORT SHOULD NOT BE RETURNED UNLESS RETURN IS REQUIRED BY SECURITY CONSIDERATIONS, CONTRACTUAL OBLIGATIONS, OR NOTICE ON A SPECIFIC DOCUMENT.

REPORT DOCUMENTATION PAGE					<i>Form Approved</i> <i>OMB No. 0704-0188</i>	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.						
1. REPORT DATE (DD-MM-YY) December 2002		2. REPORT TYPE Final		3. DATES COVERED (From - To) 02/01/1998 – 12/31/2002		
4. TITLE AND SUBTITLE AN OPEN LOGICAL PROGRAMMING ENVIRONMENT A Practical Framework for Sharing Formal Models				5a. CONTRACT NUMBER F30602-98-2-0198		
				5b. GRANT NUMBER		
				5c. PROGRAM ELEMENT NUMBER 62301E		
6. AUTHOR(S) Robert L. Constable Christoph Kreitz				5d. PROJECT NUMBER G356		
				5e. TASK NUMBER 01		
				5f. WORK UNIT NUMBER 01		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> Cornell University Department of Computer Science 4130 Upson Hall Ithaca, NY 14853 </div> <div style="width: 45%;"> Defense Advanced Research Projects Agency Information Technology Office 3701 North Fairfax Drive Arlington, VA 22209-2308 </div> </div>				8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Information Directorate Air Force Research Laboratory Air Force Materiel Command Wright-Patterson Air Force Base, OH 45433-7334						
10. SPONSORING/MONITORING AGENCY ACRONYM(S) AFRL/IFSC				11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-IF-WP-TM-2003-1501		
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.						
13. SUPPLEMENTARY NOTES This grant was partly funded under BAA #00-23, PROGRAM COMPOSITION FOR EMBEDDED SYSTEMS (PCES) by DARPA/ITO.						
14. ABSTRACT (Maximum 200 Words) <p>The project has designed, built and tested a prototype system called a Logical Programming Environment (LPE), which provides the means to formally specify, design, verify, and optimize distributed embedded systems.</p> <p>The LPE has been used in increasingly complex applications, ranging from automatic code improvements for the Ensemble group communication system to the formal design of adaptive network systems and the automatic generation of coordinated contracts for BBN's Unmanned Aerial Vehicle (UAV) application. In each case, using the LPE has led to significantly increased assurance, flexibility, or efficiency of the application.</p> <p>In the process, substantial extensions to the LPE's logical foundations and its automated reasoning capabilities were made, thus increasing its ability to contribute to the design and implementation of reliable, re-usable, re-configurable, correct, and efficient distributed embedded systems.</p>						
15. SUBJECT TERMS Formal verification, optimization, design, Dynamic embedded systems						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT: SAR	18. NUMBER OF PAGES 14	19a. NAME OF RESPONSIBLE PERSON (Monitor) Ronald Szkody 19b. TELEPHONE NUMBER (Include Area Code) (937) 255-4709 x3165	
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified				

An Open Logical Programming Environment: A Practical Framework for Sharing Formal Models

Cornell University

Final Report, December 2002

Objective

Commercial networking software is too unreliable and too insecure to be used in critical applications, especially in the military. This concern has been elevated to a national priority by the Presidential Commission on Critical Infrastructure Protection and by the President's Information Technology Advisory Committee (PITAC). The main project goal was to create highly innovative techniques and systems that can be integrated into the software development process to substantially improve reliability and durability of the results. We have demonstrated the applicability of our contributions on networking software, specifically on computing networks such as BBN's UAV and Boeing's BoldStroke applications.

Approach

The project has designed, built and tested a prototype system called a Logical Programming Environment (LPE). The LPE provides the means to formally specify and check properties of system design and code as it is being developed, as well as to verify and optimize code that has already been written.

The task of formally checking properties of code is organized in the LPE as an extension of static type checking. The type checker is enhanced by a theorem prover. Some properties depend upon a great deal of knowledge about a particular system architecture, such as event channels and event notification services, as well as upon general mathematical knowledge about common data structures and mathematical types. Much of the general mathematical knowledge has been formally proved by several theorem-proving systems. The LPE is designed so that this general knowledge can be shared; sharing is achieved by providing access to the libraries of various theorem provers through an LPE component called a Formal Digital Library.

The specific approach of this project proceeded simultaneously on three major areas.

- First, the logical language of the LPE was used to build formal models of networked embedded systems as well as formally verified knowledge and tailored analysis strategies.
- Second, the LPE was used to specify dynamic embedded systems by composition of services and to generate re-usable, re-configurable, correct, and reliable code for them, thus increasing the assurance, flexibility, and efficiency of key applications. After 2000 this was focused especially on the DARPA OEP.
- Finally, the capabilities of the LPE were continuously enhanced by extending its logical language and by integrating new automatic reasoning techniques that support the verification of embedded networked systems as well as reasoning about program composition, property-preserving code transformations, and real-time aspects.

Accomplishments

In the course of the project we have successfully applied the Logical Programming Environment in increasingly complex applications, ranging from formal support for the Ensemble group communication system to the automatic generation of coordinated contracts for BBN's Unmanned Aerial Vehicle (UAV) Open Experimental Platform (OEP) and an interface between this OEP and the Logical Programming Environment. Early in the process there were significant extensions to the LPE's logical foundations and its automated reasoning capabilities.

The following gives a comprehensive summary of the specific accomplishments. An extensive account of the research results is given in the publications of the research team, which are listed below and referred to in the text.

Optimization and Verification of Communication Systems

Using the first prototype of the LPE, we have developed fully automatic tools for improving the code of the Ensemble group communication system [2,5,7,9,30]. The improved code operates three to ten times faster than the original and is generated in a matter of seconds. Comparable improvements done by hand took months of tedious and complex work on smaller examples, and the complexity led to errors in the faster code. In contrast, the code modifications created by the automatic tools are guaranteed to be correct, that is, the improved code computes the exact same results as the original.

We have rigorously proved safety properties of the total ordering layer of Ensemble (ETO) using IO automata, and we used the proof to guide correction of a subtle error in that layer [6,8]. The proof also led to the proper repair of the error.

Formal Design of Adaptive Systems

We have designed a generic switching protocol for the construction of adaptive network systems [16] and formally proved it correct with the Logical Programming Environment [17,20,21]. In the process we have developed a formal characterization of communication properties that can be preserved when the system switches between different protocols. We have also developed an abstract characterization of invariants that have to be satisfied by an implementation of the switching protocol in order to work correctly.

As foundation for this work we have introduced the novel concept of meta-properties. Meta-properties make it possible to give an abstract characterization of "switchable" system properties, which in turn makes it easier to check whether a specific set of protocols can be employed in an adaptive system. We have described switchable properties in terms of several meta-properties such as "safety", "asynchrony", "delayable", and "send-enabled", as well as "composability" and "memorylessness". The first four of these properties are required for any layered communication system while the latter are necessary for switching. The abstract approach represents a major increase in our formal understanding of distributed systems and makes it possible to support the formal analysis and design of networked systems, including those dealing with real-time and embedded systems

With the LPE we have formally proven that communication properties that satisfy these six meta-properties are preserved under switching, whenever the switch maintains a simple synchronization invariant. The verification efforts revealed a variety of implicit assumptions that are usually made when designing communication systems and uncovered minor design errors that would have otherwise made their way into the implementation. This demonstrates that formal reasoning about group communication in an expressive theorem proving environment such as the Logical Programming Environment can contribute to the design and implementation of verifiably correct network software.

We have evaluated the performance implications of using our hybrid protocol by switching between two well-known mechanisms for implementing total order and shown that switching close to the cross-over point of these protocols's performance leads to the best practical results.

Knowledge-based Generation of Coordinated Contracts

We have developed and implemented a prototype of MediaNet [25], a general infrastructure for real-time network computations. MediaNet generalizes the computing network underlying both BBN's UAV applications and parts of Boeing's BoldStroke architecture.

In this setting we have developed a self-adaptive task allocation manager that controls the processing of real-time media over a network through coordinated local schedules. It is able to adapt, in user-specified ways, to changing workloads and network conditions, attempting to deliver specified quality of service and to meet other specifications. The task allocation manager assigns, based on current resource availability, computing and communication tasks to each node of the network in a way that maximizes a combination of user utility and network utilization. This functionality has been demonstrated in a configuration where we throttle network bandwidth and show how the system adapts between different compression schemes in order to maintain a smooth video transport. It can serve as Resource Allocation Manager for the BBN's UAV system and also provides a setting for formal reasoning about aspect-oriented design and code assembly.

Building upon the abovementioned work on specifying, verifying, and formally designing communication protocols, we have developed a formal model of networked stream computations that allows us to incorporate both real-time constraints and resource limitations into our specifications. The model makes it possible to reason about safety and liveness properties and about self-adaptation wrt. different schedules and changing data formats. It also makes it possible to factor UAV computations into aspects, including functional requirements, QoS requirements and security requirements. We can also weave fault tolerant communications into the distribution of operations on the underlying computing network.

Within the Logical Programming Environment we have, using the above formal model, developed an algorithm that derives coordinated local schedules and quality of service contracts from a global schedule. The algorithm approximates the behavior of the global scheduler with a distributed collection of local schedulers, one for each network node. A local scheduler assigns tasks to its node based only on its observation of the bandwidth of its output links, its cpu resources, and information, called "tags", passed to it by its predecessor nodes.

To generate the local schedulers, we use the MediaNet global scheduler as an offline "black box". The key idea of the algorithm is to use the logical form of the user specifications to create the set of tags. The tags correspond to all the subterms of the user specs. Proceeding in topologically sorted order, at each node we know the combinations of tags produced by the predecessor nodes. For each combination of tags we can compute a new user specification that replaces the subterms

corresponding to the tags by the input streams of the node. Using this new specification we call the global scheduler with varying bandwidth parameters and tabulate the tasks that it assigns to the current node. In this part of the algorithm we use a machine learning method called "support vector machines". In this way we create a table of tasks to assign to the current node as a function of the input tags and the output bandwidths, and this is the specification of the local scheduler.

Integration into Open Experimental Platforms

To support formal reasoning about media computing networks such as MediaNet and BBN's UAV applications we have implemented an XML interface for our LPE. It makes it possible to automatically import XML specifications generated by the MediaNet scheduler or by Vanderbilt's graphical modeling (GME) tools into our LPE and to formally analyze the actual code of MediaNet and the XML representation of BBN's UAV provided by Vanderbilt's GME.

We have developed prototypical techniques for translating the schedules generated by the LPE and MediaNet into a representation suitable for the GME and used them to automatically create coordinated CDL contracts. Ongoing contacts with Vanderbilt and BBN will enable us to refine these techniques such that the generated CDL contracts can be automatically deployed to BBN's UAV network.

Logical Foundations

We have developed a formal class theory that provides a logical foundation for design and verification through composition and weaving [15,27,29]. The theory provides the logical laws of records, modules, subtyping, and objects as well as operations for composing modules and properties. Our formal intersection operator can be used to express both functional composition and aspect weaving, and is guaranteed to combine all safety properties of the composed code pieces. Class theory is therefore well-suited as logical foundation for compositional design and verification.

We also have developed a theoretical basis for an efficient logical reflection mechanism [28]. It will enable the LPE to analyze intensional properties of systems such as the computational complexity [18,22,23] of generated software, as well as timing, use of resources, or synchronization.

Tools for Automated Reasoning and Formal Documentation

We have significantly enhanced the automatic reasoning tools of the LPE by adding generic proof techniques that support the verification of networked systems and their implementations and proof strategies especially tailored towards reasoning about program composition, aspect weaving, and embedded systems. Substantial new reasoning capabilities are now in place.

We have integrated JProver [3,12,13,19], a fully automated theorem prover for constructive first-order logic, as an external proof engine into the LPE. JProver operates on matrices and connections, a very compact representation of the search space that substantially reduces the time needed for finding proofs. Extensions of Jprover towards inductive theorem proving have been explored in theory [1,4,11,24] and are currently being added to the theorem prover.

We have introduced new techniques for asynchronous and parallel theorem proving [10,26], and are currently adding strategies that utilize external proof systems such as PVS and MetaPRL [14] as well as constraint solvers and computer algebra systems.

We have implemented tools that enable the verification system to learn from the work already done by "mining" proofs for reasoning steps that can be reused as "derived inference rules".

We have developed mechanisms for the creation of formal documentation in the LPE. Documentation with references to the actual LPE proofs are now part of the persistent LPE library and thus accessible to search and dependency tracking mechanisms. They can be viewed online or converted into a typeset version for publication.

The use of these techniques has significantly increased the degree of automation in formal design and verification and will increase the productivity of rigorous design methods.

Publications

1. B. Pientka and C. Kreitz. Instantiation of Existentially Quantified Variables in Inductive Specification Proofs, 4th International Conference on Artificial Intelligence and Symbolic Computation, LNAI 1476, pp. 247-258, Springer, 1998.
2. C. Kreitz, M. Hayden, J. Hickey. A Proof Environment for the Development of Group Communication Systems, 15th International Conference on Automated Deduction, LNAI 1421, pp. 317-332, Springer, 1998.
3. C. Kreitz and J. Otten. Connection-based Theorem Proving in Classical and Non-classical Logics, Journal for Universal Computer Science 5(3):88-112, Springer-Verlag, 1999.
4. B. Pientka and C. Kreitz. Automating inductive Specification Proofs, Fundamenta Informatica 39(1-2):189-209, 1999.
5. C. Kreitz. Automated Fast-Track Reconfiguration of Group Communication Systems, 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, LNCS 1579, pp. 104-118, Springer, 1999.
6. J. Hickey, N. Lynch, and R. van Renesse. Specifications and proofs for Ensemble layers. 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, LNCS 1579, pp. 119-133, Springer, 1999.
7. X. Liu, C. Kreitz, R. van Renesse, J. Hickey, M. Hayden, K. Birman, R. Constable. Building Reliable, High-Performance Systems from Components, 17th ACM Symposium on Operating System Principles (SOSP'99), Operating Systems Review 34(5):80-92, 1999.
8. Mark Bickford and Jason Hickey. Predicate transformers for infinite-state automata in Nuprl type theory. In Irish Formal Methods Workshop, 1999.

9. K. Birman, B. Constable, M. Hayden, J. Hickey, C. Kreitz, R. van Renesse, O. Rodeh, W. Vogels. The Horus and Ensemble Projects: Accomplishments and Limitations. DARPA Information Survivability Conference and Exposition (DISCEX 2000), Hilton Head, SC, pp. 149-160, IEEE Computer Society Press, 2000.
10. S. Allen, R. Constable, R. Eaton, C. Kreitz, L. Lorigo. The Nuprl Open Logical Environment, 17th International Conference on Automated Deduction, LNAI 1831, pp. 170-176, Springer, 2000.
11. C. Kreitz, & B. Pientka. Matrix-based Inductive Theorem Proving, International Conference TABLEAUX-2000, LNAI 1847, pp. 294-308, Springer, 2000.
12. C. Kreitz, J. Otten, S. Schmitt, B. Pientka. Matrix-based Constructive Theorem Proving. In *Intellectics and Computational Logic. Papers in honor of Wolfgang Bibel*, pp. 189-205, Kluwer, 2000.
13. C. Kreitz and S. Schmitt. A Uniform Procedure for Converting Matrix Proofs into Sequent-Style Systems, *Journal of Information and Computation* 162(1-2):226-254, 2000.
14. J. Hickey and A. Nogin. Fast Tactic-based Theorem Proving. 13th International Conference TPHOLS 2000. *Lecture Notes in Computer Science* 1869, pp. 252-266, Springer Verlag, 2000.
15. R. Constable and J. Hickey. Nuprl's Class Theory and its Applications. *Foundations of Secure Computation*, NATO ASI Series, Series F: Computer & System Sciences, pages 91-116. IOS Press, 2000.
16. X. Liu, R. van Renesse, M. Bickford, C. Kreitz, R. Constable. Protocol Switching: Exploiting Meta-Properties, *International Workshop on Applied Reliable Group Communication (WARGC 2001)*, pp. 37-42. IEEE CS Press, 2001.
17. M. Bickford, C. Kreitz, R. van Renesse, X. Liu. Proving Hybrid Protocols Correct, Technical Report, Cornell University, Ithaca, NY, February 2001.
18. R. Benzinger. Automated Complexity Analysis of Nuprl Extracted Programs. *Journal of Functional Programming* 11(1):3-31, 2001
19. S. Schmitt, L. Lorigo, C. Kreitz, A. Nogin. JProver: Integrating Connection-based Theorem Proving into Interactive Proof Assistants, *International Joint Conference on Automated Reasoning*, LNAI 2083, pp. 421-426, Springer Verlag, 2001.
20. M. Bickford, C. Kreitz, R. van Renesse, R. Constable. An Experiment in Formal Design using Meta-Properties, *DARPA Information Survivability Conference and Exposition II (DISCEX 2001)*, pp. 100-107, IEEE Computer Society Press, 2001.
21. M. Bickford, C. Kreitz, R. van Renesse, X. Liu. Proving Hybrid Protocols Correct, 14th International Conference on Theorem Proving in Higher Order Logics, LNCS 2152, pp. 105-120, Springer Verlag, 2001.

22. R. Constable and K. Crary. Computational complexity and induction for arithmetically computable functions in type theory. *Reflections on the Foundations of Mathematics: Essays in Honor of Solomon Feferman*, Lecture Notes in Logic, pp 166-183. Association for Symbolic Logic, 2001.
23. A. Nogin. Writing Constructive Proofs Yielding Efficient Extracted Programs. *Electronic Notes in Theoretical Computer Science* 37, 2001.
24. C. Kreitz, & B. Pientka. Connection-based Inductive Theorem Proving, *Studia Logica* 69(2):293-326, 2001.
25. M. Hicks, R. van Renesse, M. Bickford, R. Constable, C. Kreitz, L. Lorigo. User-specific Adaptive Scheduling in a Streaming Media Network, Technical Report, Cornell University, Ithaca, NY, 2002.
26. S. Allen, M. Bickford, R. Constable, R. Eaton, C. Kreitz, L. Lorigo. FDL: A Prototype Formal Digital Library, Technical Report, Cornell University, Ithaca, NY, 2002.
27. A. Kopylov. Representation of object calculus in type theory, Technical Report, Cornell University, Ithaca, NY, 2002.
28. E. Barzilay and S. Allen. Reflecting Higher-Order Abstract Syntax in Nuprl. 15th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2002), pp. 23-32, NASA, 2002.
29. R. Constable. Naive Computational Type Theory. *Proof and System-Reliability*, NATO Science Series III, pp 213-260, 2002.
30. C. Kreitz. Building Reliable, High-Performance Networks with the Nuprl Proof Development System, *Journal of Functional Programming*, 2003 (to appear).